

# Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs

Mary Ann Lapham  
Michael Bandor  
Eileen Wrubel

**January 2014**

**TECHNICAL NOTE**  
CMU/SEI-2013-TN-031

**Software Solutions Division**

<http://www.sei.cmu.edu>

Copyright 2014 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

This report was prepared for the  
SEI Administrative Agent  
AFLCMC/PZM  
20 Schilling Circle, Bldg 1305, 3rd floor  
Hanscom AFB, MA 01731-2125

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution except as restricted below.

Internal use: \* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

External use: \* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

\* These restrictions do not apply to U.S. government entities.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0000792

---

## Table of Contents

<b>Acknowledgments</b>	<b>vii</b>
<b>Executive Summary</b>	<b>ix</b>
<b>Abstract</b>	<b>xiii</b>
<b>1      Introduction</b>	<b>1</b>
1.1    Approach	1
1.2    Contents of This Document	2
<b>2      “New” View of Acquisition Life Cycle</b>	<b>3</b>
2.1    Current Acquisition Life Cycles	3
2.2    Acquisition Life Cycle vs. Software Development Life Cycle	5
2.3    Sample Approach to Harmonizing the Software Development Life Cycle and the Acquisition Life Cycle Framework	7
2.4    Addressing the Differences in the Life Cycle Milestones and Reviews	9
<b>3      Performing Effective Technical Evaluation in the Iterative/Agile Environment</b>	<b>15</b>
3.1    Technical Evaluation Participants	15
3.2    Cultural Adaptation to Iterative Methods	15
3.3    Collaborative Environment	17
3.4    Common Areas of Contention: Systems Engineering and Test	19
3.4.1    Evaluating Systems Engineering Estimates	20
3.4.2    Evaluating Test Engineering Estimates	20
3.4.3    Using Actual Data as a Basis of Estimate	21
3.5    Oversight and Insight Implications	22
<b>4      Assessment of Contractor Estimating Methodology</b>	<b>24</b>
4.1    Methodology Documentation Review	24
4.1.1    Work Breakdown Structure and Skill Mix	25
4.1.2    Estimating Methodology	26
4.1.3    Additional Resources	28
<b>5      Conclusion</b>	<b>31</b>
<b>Appendix A   Evaluation Question Checklist</b>	<b>32</b>
<b>Appendix B   Acronyms</b>	<b>35</b>
<b>References/Bibliography</b>	<b>37</b>



---

## List of Figures

Figure 1: DoD Defense Acquisition Management System - Single Step Acquisition (2008)	3
Figure 2: DoD Defense Acquisition Management System - Evolutionary (2008)	3
Figure 3: BCL Based Acquisition Framework (2011)	5
Figure 4: Acquisition Life Cycle and Software Development (Example)	6
Figure 5: Program Decomposition	7
Figure 6: Program Decomposition With Major Reviews	8



---

## List of Tables

Table 1: Progressive Development Activities	9
Table 2: PMO Options Supporting Iterative/Incremental Development	11
Table 3: Comparison of Iterative with Traditional DoD Cultural Elements	16
Table 4: Questions to Ask When Evaluating Iterative Proposals	32



---

## Acknowledgments

The authors wish to thank Mr. Blaise Durante, the former Air Force Deputy Assistant Secretary for Acquisition Integration (now retired), for his continued support of SEI and this project. This support allowed the authors to produce a third report addressing another topic of interest to DoD acquisition offices and development organizations that are currently pursuing or are contemplating pursuing acquisition strategies that employ one or more elements of a set of incremental development methods commonly termed “Agile methods.”

In addition, the authors would like to express our appreciation for all those who reviewed this technical note. Your insights were invaluable. We extend our sincerest thanks to the following SEI people:

Robert Ferguson

John Foreman

Harry Levinson

Suzanne Miller

William Novak

David Zubrow



---

## Executive Summary

Today's program management office (PMO) realizes that its typical methods for evaluating technical proposals resulting from Requests for Changes (RFCs) are based on traditional government methods for independent cost estimation. These traditional methods are not well suited to provide a relevant evaluation of contractor estimates based on an iterative or agile development approach. Iterative development estimation uses a just-in-time approach which tends to start with a high-level relative estimate that is refined to create detailed absolute estimates as more is learned about the operational context and user needs. Traditional methods typically employ mainly absolute estimates from the beginning. Thus, the expected amount of detail usually provided by traditional methods early in the negotiation of RFCs may be missing in iterative estimates.

This technical note is the product of case studies of actual Department of Defense (DoD) programs (program and contractor identifying information have been redacted) whose contractors leverage agile methods. The information was gathered during our review of Requests for Change (RFCs), reports, and contractor CDRLs and through discussions held with Program Management Office (PMO) staff to gain an understanding of the challenges and miscommunications that can occur when evaluating technical RFC proposals in an agile environment. Many of the challenges faced were not necessarily *unique* to agile methods. However, the problem solving approach to these issues was complicated by the mismatch of expectations by the PMO who were coming from deep waterfall backgrounds and the contractor who had made the shift to the agile software development methodology. This technical note describes challenges uncovered on the review of Request for Changes (RFCs) after the beginning of program execution.

Confusion often occurs when the program office tries to interpret iterative activities relative to the traditional DoD acquisition life cycle framework. There seems to be a common misunderstanding that the software development methodology must mirror the acquisition life cycle. From a statutory perspective, the acquisition life cycle is, in fact, software methodology "agnostic;" it does *not* prescribe a traditional waterfall-based engineering approach. Problems occur when trying to overlay "traditional" acquisition milestone events directly atop software development methodologies that utilize smaller and more numerous work units, without understanding the relationship between those work units and the milestone events. This is where the program office needs to be very aware of the execution of the development method and the differences that iterative development creates.

There are three possible approaches to synchronizing traditional milestones with iterative development. They are

- The PMO uses the major milestone events (e.g., PDR and CDR) in each block as traditional milestone events (little acknowledgement of iterative concepts).
- The PMO participates in each progressive review throughout the iteration (great acknowledgement of iterative concepts).
- The PMO has technical staff participate in each progressive review and the major milestones become a management level review (a hybrid approach).

Each approach has advantages and disadvantages. Selected advantages and disadvantages will be addressed in this technical note.

A key finding in reviewing the technical evaluation process is that many of the same tasks executed during traditional estimation evaluation of RFCs still need to be performed when evaluating estimates created using iterative methods. The focus of the evaluation will be slightly different and will necessarily take into account the maturity of each iteration and the associated software and documentation. This shift in perspective is required to align the review with the shift in the underlying culture or environment created by the iterative method.

The organizational structure, leadership style, rewards system, communication models, decision making models, and staffing models are likely to be different in an iterative environment versus a traditional DoD environment. Each of these elements still needs to be addressed but the context will vary depending on the environment. For instance, communication in an iterative environment includes a variety of interim reviews that focus on just the software being developed in that iteration, as opposed to the large reviews seen in the traditional environment where the entire project is the focus. Additionally, documentation in the iterative environment tends to be user and maintainer-focused and represents “just enough” to promote understanding, with the overall documents maturing as they progress through successive iterations (“just enough” will vary from situation to situation depending on the needs and regulation requirements of the project [Lapham 2010]). In the traditional environment, drafts of documents are produced for the entire project (that attempt to be complete prior to ensuing tasks), are labored upon in significant detail, and then revised and refined as the project progresses. Some of these are obviated when the software is actually produced, and all of them are subject to becoming obsolete as more is learned about the operational context and actual operational need that informs the software acquisition.

While some level of collaboration between the program office, contractor(s), and operators is required on any program, iterative styles expect a higher level of collaboration. This collaboration consists of open, candid communications with a significant amount of face-to-face interaction among multiple stakeholders across the development, program office, and user communities. Informed participation by the government in interim or progressive reviews—as well as at major milestones—is particularly important, because this intense communication doesn’t take place for its own sake. It occurs as part of building and sustaining trust across the three communities.

In order to understand the contractor approach to iterative estimation, the government program office typically reviews the contractor’s Basis of Estimate documentation as well as related documents that are involved in a software-intensive acquisition, including the Software Development Plan (SDP), the System Engineering Management Plan (SEMP), and the Master Software Build Plan (MSBP). These documents provide significant information about the contractor’s iterative approach and estimation process. However, be aware that the documents are likely written assuming that the reader has a complete understanding of iterative principles. In this case, the program office should look in several additional areas where information can provide government evaluators with greater insight into the contractor’s estimation process.

For example, many software-centric iterative methodologies seem to consider subsystem and segment engineering hours as outside the scope of their typical iteration process and thus apply an adjustment factor to their development hours to obtain the system engineering hours. While this is an acceptable practice, if the RFC technical estimate affects work already done in previous itera-

tions, the government should expect some rework in systems engineering. However, if the RFC technical estimate is for work not yet performed, then the scope of the new work needs to be evaluated to determine if system engineering hours need to increase or decrease.

Overall, effective evaluation of RFC technical estimates that come out of an iterative development approach requires that the evaluators understand the principles of communication, learning, and trust that are at the core of most iterative approaches using agile methods. This is not meant to imply that the statement “trust us, we’re doing Agile” should be taken at face value. However, it does mean that setting expectations about the progressive maturity of technical, programmatic, and working software deliverables is required.



---

## Abstract

This technical note is the third in an SEI series on the adoption of lean and agile methods in the DoD. Agile topics in acquisition were introduced in CMU/SEI-2010-TN-002 and CMU/SEI-2011-TN-002. This technical note extends the topics covered into the evaluation and negotiation of technical proposals that reflect iterative development approaches that in turn leverage agile methods. This framework is intended for use by government program office personnel who seek to understand evaluation approaches in this context. The information and recommendations contained in this report result from observations of defense acquisition programs wherein contractors employed iterative methods such as Agile software development methodology (hereafter referred to as “agile”). Key questions for discussion with the contractor are provided, along with agile perspectives on why certain items will be defined differently depending on whether the contractor is using agile or iterative methods for software development. The intended audience for this paper includes any government personnel who need to support or participate in negotiations with contractors for changes to the contract that is in place to develop software using agile or iterative methods.



---

# 1 Introduction

Typically, the government creates an independent cost estimate for each technical proposal received in response to a Request For Change (RFC)<sup>1</sup> on an active development contract. This estimate is used to determine a fair and reasonable cost for the proposed work. It is used during negotiations of the final price. However, if the methodology being employed by the contractor for these software intensive proposals is based on agile or other iterative methods, the traditional government method for independent cost estimation becomes inappropriate as the basis for a relevant comparison. This is not to say that the government responsibilities to evaluate the technical proposals will change. Rather, how the government executes those responsibilities may change, as it can be impacted by the methodology used by the contractor. Remember that the goal is a negotiated agreement as to the overall cost of the technical proposal for the RFC; how you achieve that goal when using iterative or agile methods is the variation discussed in this paper.<sup>2</sup>

## 1.1 Approach

Given the ever-increasing possibility that contractors will be employing methods other than the traditional waterfall approach, how can the government effectively evaluate any proposed offering? This technical note will address the following issues:

1. Identify what questions should be answered to verify that the contractors costing methodology is clear and complete: This information will allow the government personnel to do a more thorough review and conduct more effective and reasonable negotiations.
2. Develop an alternate method or roadmap for government use that is compatible with the contractor's so that the government can effectively estimate costs and evaluate proposals. This alternate method will identify contractor information the government needs to review in order to understand the proposed change. It will allow the government to develop its own independent cost estimate for the technical proposal.

The proposed alternative methodology is based on general knowledge of the concepts underlying the agile or iterative methods employed by contractors and a general review of some Requests for Change.

This technical note is based on both a literature review and interviews with diverse programs and practicing Agilists as discussed in Lapham [Lapham 2010, Lapham 2011].<sup>3</sup>

---

<sup>1</sup> Another common term for RFC is Engineering Change Proposal (ECP). For purposes of this paper we will use RFC.

<sup>2</sup> While this data might also be used to help understand evaluating responses to Requests for Proposals that involve agile or other iterative methods, no attempt was made to review or do a thorough analysis of data in that situation. The data for this paper is based upon information from programs that had awarded contracts that used agile or other iterative methods and were at the point of reviewing RFCs against those contracts.

<sup>3</sup> Identifying information withheld at customer request.

## **1.2 Contents of This Document**

The organization and intended audience for each topic in this technical note is provided below.

Executive Summary contains highlights of this document and will be beneficial for most reader types to review.

Section 1, Introduction, describes the problem of adapting government cost estimation review to change proposals on programs using agile methods. All readers should review this section.

Section 2, “New” View of Acquisition Life Cycle, describes the ways programs can interpret an iterative development project relative to the DoD acquisition life-cycle framework set forth in DoDI 5000.02. This section also addresses the popular misconception that software development methodologies must mirror the program’s acquisition life cycle. This section is intended for all reader types.

Section 3, Performing Effective Technical Evaluation in the Iterative/Agile Environment, discusses the role of technical evaluations in agile programs and the cultural shifts program offices must make in order to effectively employ agile methods within the DoD regulatory environment. This section includes juxtapositions of agile executions with the acquisition life cycle. The audience for this section includes policy makers, program managers, and program teams.

Section 4, Assessment of Contractor Estimating Methodology, provides discussion and guidance for program offices for the purpose of assessing the validity of contractor estimation models during the evaluation of contractor proposals resulting from RFCS. Its primary audience is acquisition program managers and staff involved in the technical evaluation process.

Section 5, Conclusion, provides a brief summary of the topics addressed in this paper. This section is intended for all reader types.

Appendix A, Acronyms, defines acronyms we use in the main body of the report.

Appendix B, Evaluation Question Checklist, provides a list of helpful questions for program managers to apply to contractor estimates.

## 2 “New” View of Acquisition Life Cycle

As the old joke goes, “How do you eat an elephant? One bite at a time!” The general approach in an agile or iterative development project is to take the software capabilities and requirements and implement them via smaller “bites” (work units) within the overall system development framework. Problems arise when interpreting iterative development activities relative to the traditional DoD acquisition life-cycle framework.

### 2.1 Current Acquisition Life Cycles

In the current revision of DoDI 5000.02, programs have two options for acquiring systems: a single-step acquisition or an evolutionary acquisition performed in increments [DoD 2008]. These alternatives are depicted in Figure 1 and Figure 2 below. See Appendix A for definitions of the acronyms shown in Figure 1 and Figure 2.

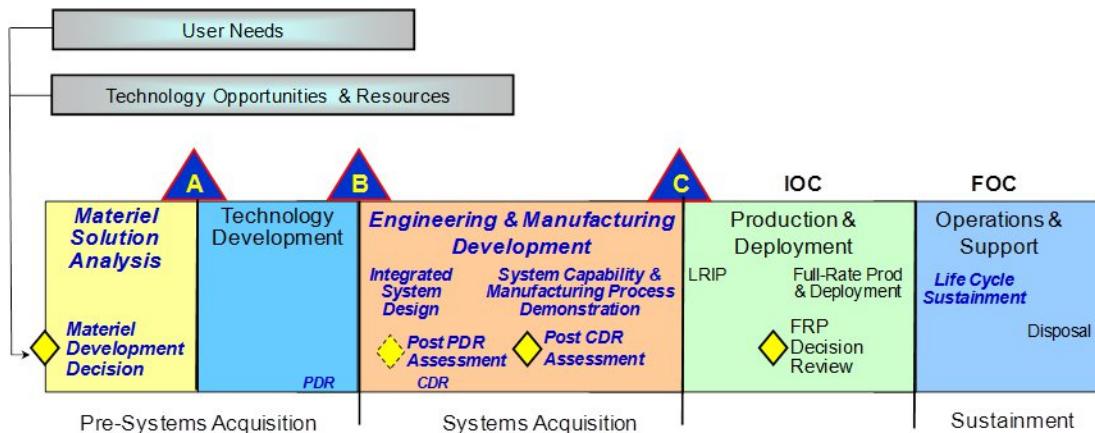


Figure 1: DoD Defense Acquisition Management System - Single Step Acquisition (2008)

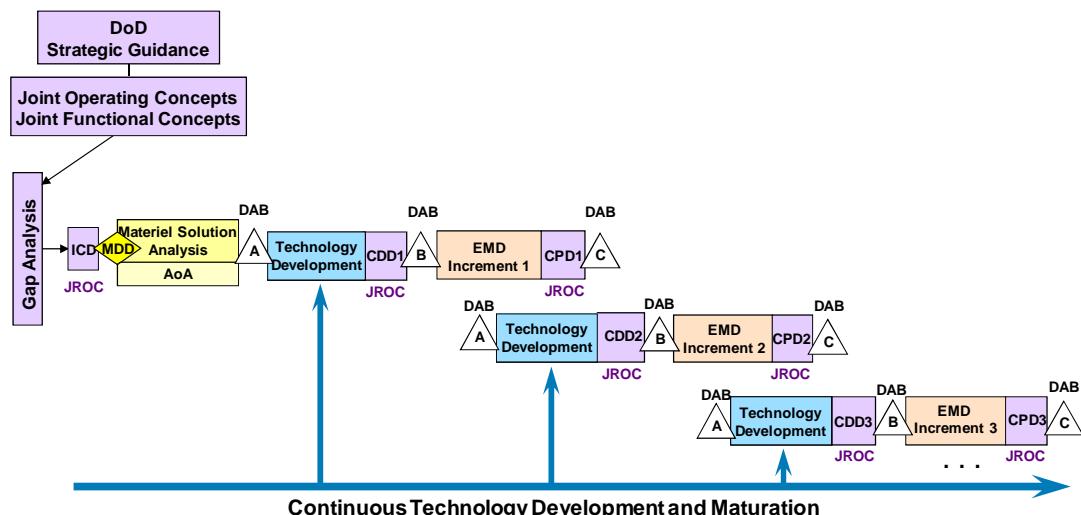


Figure 2: DoD Defense Acquisition Management System - Evolutionary (2008)

The Defense Acquisition Guide (DAG) provides further information regarding the evolutionary acquisition strategy [DAU 2011]:

*Evolutionary acquisition strategies integrate advanced, mature technologies into producible systems that can be deployed to the user as quickly as possible. An evolutionary acquisition strategy matches available technology and resources to approved, time-phased, incremental delivery of capability needs. Systems engineering processes provide the disciplined, integrated development and production environment that supplies increasing capability to a materiel solution. In incremental development, capability is developed and fielded in increments with each successive increment building upon earlier increments to achieve an overall capability. These approaches to evolutionary acquisition are particularly effective in quickly fielding an initial capability or increment of functionality while allowing continued efforts to incrementally attain the final, full, end-state capability. Robust systems engineering processes ensure that systems are designed to easily and affordably accommodate additive capabilities in subsequent increments (e.g., modular, open systems design). The incremental development relies heavily on prototyping, both physical and functional, to get stakeholder feedback and reduce risk.*

*Evolutionary acquisition has increased the importance of traceability in program management. If a defense system has multiple increments, systems engineering can trace the evolution of the system. It can provide discipline to and documentation of the repeated trade-off analyses and decisions associated with the program. Because of the nature of evolutionary acquisition, design, development, deployment, and sustainment can each be occurring simultaneously for different system increments.*

As part of the efforts to address acquisition of Information Technology systems, the DoD is adopting an additional life cycle based on the Business Capability Lifecycle (BCL) [DoD 2011a, DoD 2011b]. This approach utilizes iterations and does not generally involve hardware development. An example of this new approach is shown in Figure 3. See Appendix A for a listing of the acronyms in the figure.

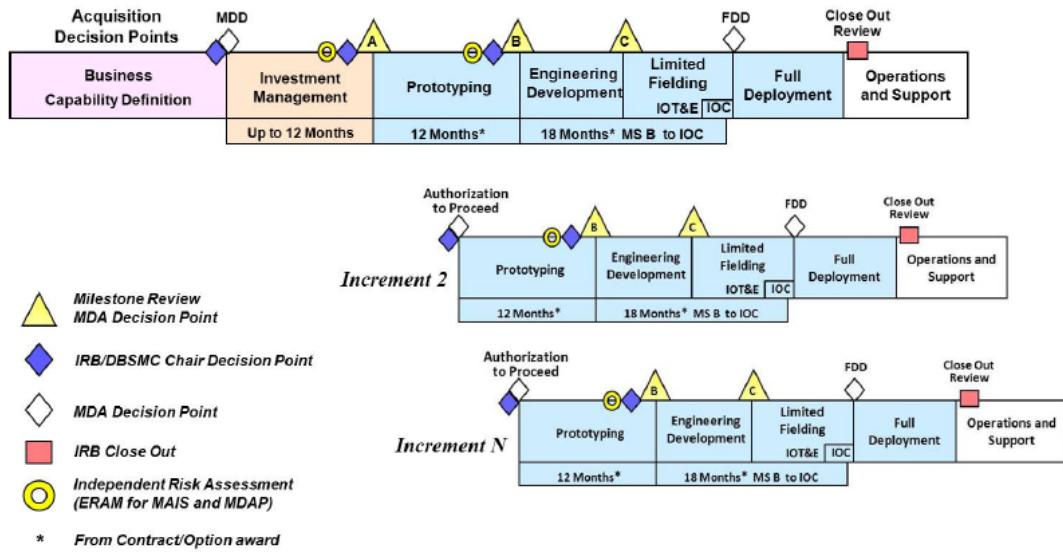


Figure 3: BCL Based Acquisition Framework (2011)

In the latter examples, the approach delivers a useful capability at the end of each increment. A point that is missed quite often is that these acquisition frameworks are not the same as life cycles represented in software development methodologies.

## 2.2 Acquisition Life Cycle vs. Software Development Life Cycle

There is a common misunderstanding that the software development methodology in use on a program must mirror its acquisition life cycle. The acquisition life cycle is, in fact, software methodology “agnostic.” Conflict may occur when attempting to apply “traditional” acquisition life cycles and milestone events to the software development methodologies that utilize smaller and more numerous work units. A traditional acquisition life cycle expects artifacts to be at a more homogeneous level of maturity than is typical in an iterative software development. A notional example showing the acquisition life cycle (as increments) and the contractor software development (combination of spiral and incremental) is shown in Figure 4.

In this fictitious example, the contractor proposed two different software methodologies to accomplish the work needed for each specific increment. The software development methodologies could have been an agile method such as XP or Scrum, as well as incremental and spiral development.

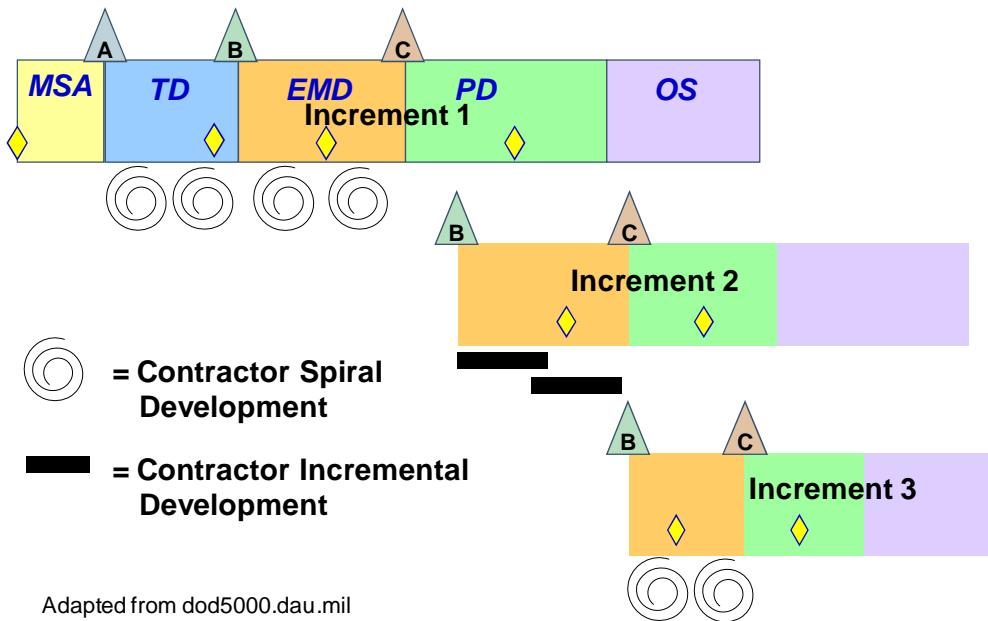


Figure 4: Acquisition Life Cycle and Software Development (Example)

At the top of the diagram, the contractor completes two spirals for each major acquisition phase. In the EMD phase, one spiral finishes just before a major milestone event within the phase. Similar patterns are reflected in Increments 2 and 3. In practical terms, this means that the contractor will be “delivering” capability in between the phases and milestones that are defined in the program’s acquisition cycle. While this may have great benefit for the development of assets like technical infrastructure, it creates a disconnect in communication of accomplishment between the contractor and the government program office and its stakeholders.

If a program is in a pre-acquisition phase (has not awarded a contract) a method for addressing the mismatch between the acquisition life cycle and software development methodology is to ask the offeror the following:

1. What software development methodology will be used?
2. What artifacts are typically created as part of that methodology, and at what points in time (and list them in the Data Accession List—DAL)?
3. What is your approach to align the “milestone” events in the methodology with the acquisition life cycle?
4. What is your approach to Earned Value (EV) when applying it to the software methodology?
5. How is software quality measured?
6. What does “done” mean? (How is completion defined?)
7. Where in your Software Development Plan (SDP) is this information documented?

If you join a program office where the contract is already in place and cannot answer the above questions by looking at the Software Development Plan, then it’s worth having a conversation with the contractor that includes answering the above questions.

### 2.3 Sample Approach to Harmonizing the Software Development Life Cycle and the Acquisition Life Cycle Framework

One sample approach to achieving a harmonized life cycle that accounts for the needs of both the developer and the acquirer could divide the work into blocks. Each block roughly corresponds to a system “increment” as shown in the previous examples (Figures 2, 3 and 4) and each block (also referred to as a segment block) is comprised of one or more software increments, similar to that shown in the center of Figure 4. Another way to think about the decomposition (and the subsequent application of the software methodology) is shown in Figure 5. It depicts, in general, the relationship between the system, the segment blocks, the iterations and the capabilities.

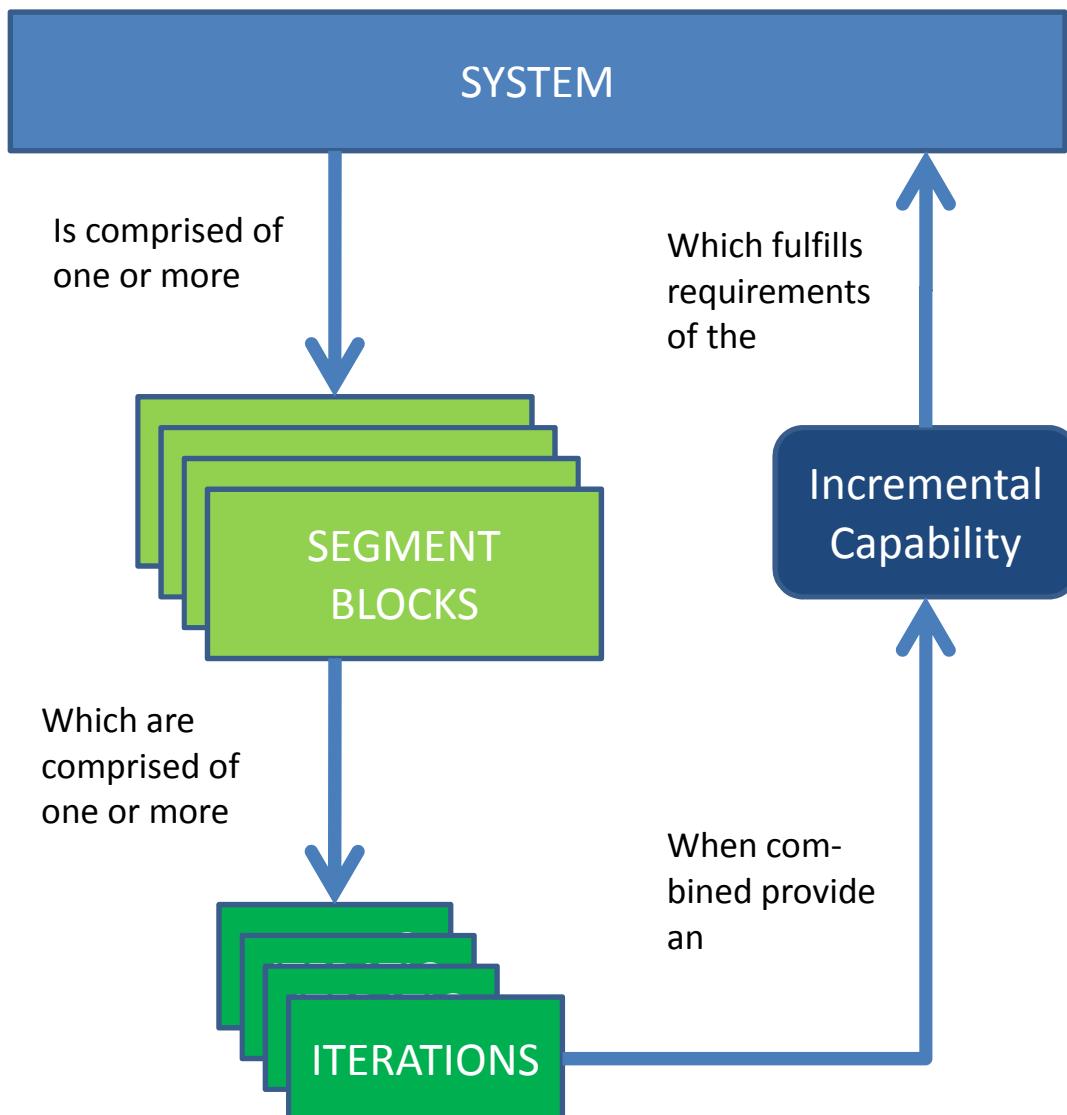


Figure 5: Program Decomposition

Using a method of decomposition like this makes the alignment of traditional software development milestones and reviews relative to the acquisition milestones a bit more challenging, particu-

larly to the government program offices. In the SDP, the contractor should identify the various milestones and reviews used to evaluate progress during and across iterations used in this development methodology. We refer to these reviews as “progressive” reviews because they occur progressively throughout the development life cycle. Adding progressive reviews to the general diagram previously discussed provides a bit more insight, as shown in Figure 6. The reviews and milestones have been added at the appropriate level based on the use of iterative reviews such as:

- Progressive Segment Design Walkthrough (PSDW)
- Progressive Preliminary Design Walkthrough (PPDW)
- Progressive Build Planning Review (PBRP)
- Progressive Critical Design Walkthrough (PCDW)
- Progressive Integration Readiness Review (PIRR)
- Progressive and Test Readiness Review (PTRR)

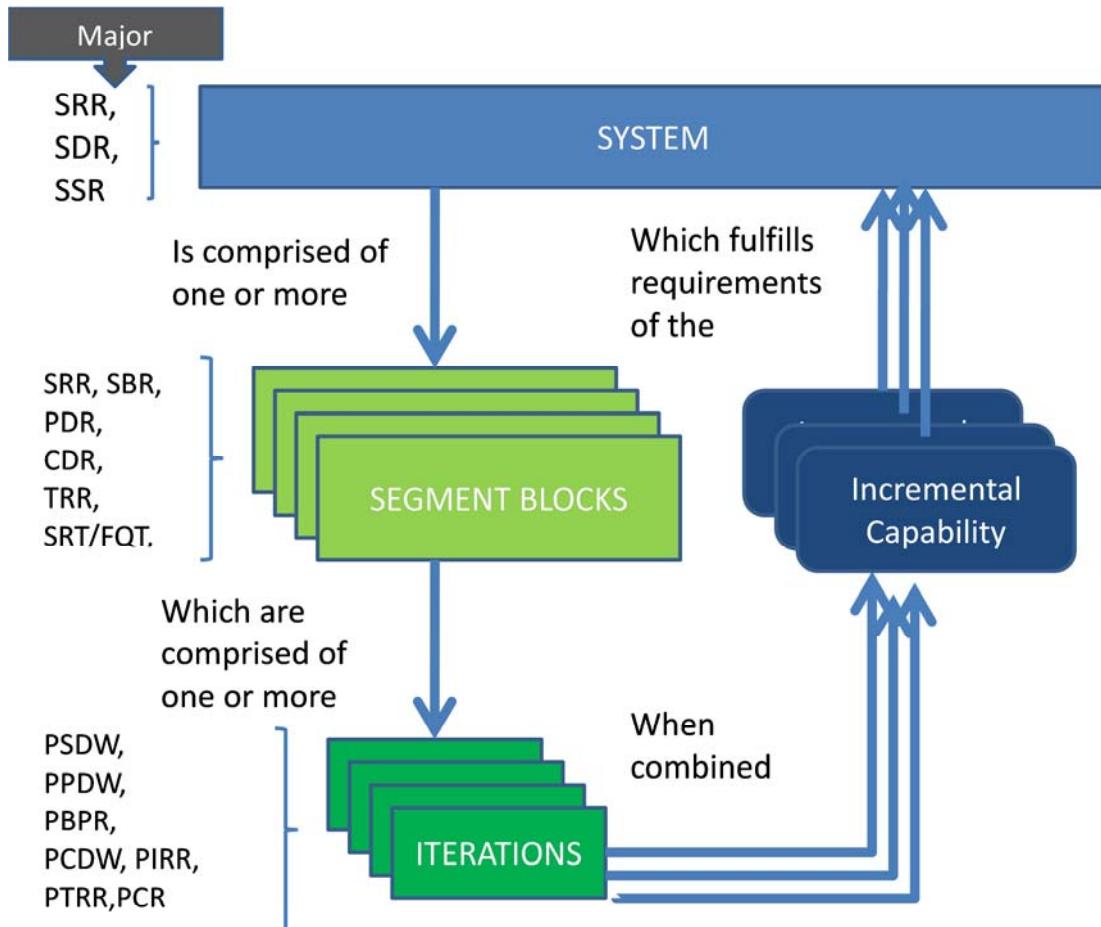


Figure 6: Program Decomposition With Major Reviews<sup>4</sup>

<sup>4</sup> Acronyms for program reviews required by DoD 5000.02 are listed and explained in Appendix B.

## 2.4 Addressing the Differences in the Life Cycle Milestones and Reviews

The contractor's incremental-iterative implementation will typically use a set number of weeks for each iteration. For illustrative purposes suppose the fictitious contractor uses a 22-week iteration<sup>5</sup> which comprises the activities and durations shown in Table 1.

Table 1: Progressive Development Activities

DEVELOPMENT ACTIVITY	DURATION	ARTIFACTS & REVIEWS
Preliminary Design and Iteration Planning	4 weeks	Engineering Artifacts, Test Plan, Progressive Preliminary Design Walkthrough (PPDW), Progressive Build Planning Review (PBPR), Actions
Detailed Design	14 weeks	Engineering Artifacts, Unit Test Plan, Progressive Critical Design Walkthrough (PCDW), Actions
Code and Unit Test (CUT) and Unit Integration and Test Software Integration and Test (SWIT)		Code & Unit Test Artifacts, Peer Reviews, Actions Progressive Integration Readiness Review (PIRR), Verify Design, Validate Software Requirement Specification (SRS) Requirements, Pre-Ship Review (PSR), Defect Work-Off
Software Configuration Item Qualification Test (CIQT) or Risk Reduction Integration and Test	6 weeks	Dry Run, Run for Record (RFR), Test Reports, Progressive Test Readiness Review (PTRR), Post Test Review (PTR), Defect Work-Off

As stated in section 2.2, the DAG provides further interpretive guidance for program managers deciding to use an incremental approach. It further discusses the issues of program reviews with this approach [DAU 2011]:

*Programs with an evolutionary acquisition strategy undergo additional reviews (e.g., a Milestone B for each increment). The systems engineering activities and reviews are repeated as appropriate to ensure the same level of program insight is achieved within evolutionary acquisition programs.*

The SEMP and SDP should provide very specific information as to when certain activities and reviews occur relative to the block (increment) as well as the iterations. This development process aligns very well with the guidance provided in the DAG. For instance, the DAG provides the following guidance for Critical Design Reviews when using an incremental approach [DAU 2011]:

*For complex systems, a CDR may be conducted for each subsystem and logistics element. These incremental reviews lead to an overall system CDR. Incremental design reviews are usually defined at Interface Control Document boundaries. System level performance is supported by compliance with Interface Control Documents, but not assured. When incremental reviews have been conducted, additional risk is introduced until the overall system CDR establishes the complete system product baseline. Each incremental CDR closes a functional or physical area of design to modification regardless of when it is held.<sup>6</sup> This completed area of design may*

<sup>5</sup> In many agile environments, iterations are much closer to 2-4 week cycles. Multiple iterations constitute a release. In this case, the contractor opted for a longer duration due to the nature of the technical work.

<sup>6</sup> In an agile approach, a functional or physical area might not necessarily be closed to modification once its CDR is complete. As with traditional development efforts, downstream changes to requirements or design may in some circumstances necessitate revisiting previously completed work. However, in agile environments reopened design and rework is part of the normal agile development plan and is not considered unusual.

*need to be reopened if open areas cannot achieve desired performance in isolation. If the schedule is being preserved through parallel design and build decisions, any system deficiencies that lead to reopening design will result in rework and possible material scrap.*

When using agile methods, the materials one might see at CDR will look different due to the iterative nature of the approach. All documentation will *not* appear at the same level of maturity:

- Some documentation will still be in draft condition (such as design documents for the overall system that support requirements that have been allocated to some future increment).
- Some documents will be partially completed (such as those supporting requirements in upcoming increments that are dependent upon the implementation of earlier capabilities).
- Some will be fully complete (perhaps for requirements that are being implemented in the current increment).

When trying to synchronize or adapt more agile software development methodologies with the more traditional acquisition life cycle, there are several options available. They are summarized in Table 2.

Table 2: PMO Options Supporting Iterative/Incremental Development

OPTION	ADVANTAGES	DISADVANTAGES
A – PMO uses the PDR and CDR events in each block as traditional milestone events	<ul style="list-style-type: none"> <li>• Fits the more traditional acquisition life cycle</li> <li>• Minimizes personnel travel costs</li> </ul>	<ul style="list-style-type: none"> <li>• Less synchronicity between development life cycle and review life cycle. CDRs and PDRs (per block) may be accomplished well into the iteration cycle raising the distinct possibility of re-work in the event there is a direction change (e.g., requirements, etc.)</li> <li>• PDR and CDR events end up being very long (3-5 days possibly) as information on each iteration will most likely be presented</li> <li>• Decreases the in-process communication between the contractors and PMO regarding development efforts</li> <li>• Impact: Artifacts under review will be of different maturity levels since various iterations will be reviewed concurrently. Some artifacts will be complete, some under construction and some not yet started. This could result in confusion to the reviewer and unnecessary contractor comment adjudication with products under construction subsequent to artifacts being made available for review. Without proper training and coordination of what's included and what is not included, this approach could be seen as defeating the purpose of PDR/CDR.</li> </ul>

OPTION	ADVANTAGES	DISADVANTAGES
B – PMO team participates in each of multiple PPDW/PCDWs (one per iteration). PDR/CDR are still held at some level of technical discussion and also includes management elements	<ul style="list-style-type: none"> <li>Allows for earlier looks at the evolving products by program office staff and end users or their surrogates</li> <li>Allows for direction change if needed much earlier in the delivery cycle of the increment</li> <li>Potentially shortens the PDR and CDR to a high-level review (summarization of development efforts and outstanding action items)</li> <li>Allows for better communication between contractors and PMO regarding development efforts</li> <li>Aligned well with the DAG guidance on incremental development</li> <li>Potentially increased synchronicity between development life cycle and review life cycle.</li> <li>Impact: Only review artifacts for the iteration being reviewed. Artifacts under review should be relatively of the same maturity level. The maturity levels need to be set by the appropriate entry and exit criteria. Depending on the situation, the artifacts could be of differing maturity but that could cause more confusion. Using the same level of maturity could result in less confusion to the reviewer and reduction of unnecessary contractor comment adjudication with few products under construction subsequent to artifacts being made available for review.</li> </ul>	<ul style="list-style-type: none"> <li>Could require more travel (if not done remotely) and resource allocation to review activities by PMO personnel; however, costs related to these activities are highly likely to result in lower overall program cost and risk</li> <li>Potential for loss of big picture view due to dependencies across iterations. The work required to maintain the big picture could be greater than for traditional approaches.</li> <li>Risk that required replanning may not receive appropriate emphasis as contents and requirements move from iteration to iteration.</li> </ul>

OPTION	ADVANTAGES	DISADVANTAGES
<p>C – PMO technical staff (engineers) participates in each PPDW/PCDW (per iteration) and PDR/CDR becomes a management level review. No technical detail is discussed other than a summary for management.</p>	<ul style="list-style-type: none"> <li><i>Potentially shortens the CDR to a high-level review (summarization of development efforts and outstanding action items)</i></li> <li>Allows for earlier looks at the evolving products by program office staff and end users and their surrogates</li> <li>Allows for direction change if needed much earlier in the delivery cycle of the increment</li> <li>Allows for better communication between contractors and PMO regarding development efforts</li> <li>Aligned well with the DAG incremental guidance</li> <li>Potentially increased synchronicity between development life cycle and review life cycle.</li> <li>Impact: Only review artifacts for the iteration being reviewed. Artifacts under review should be relatively of the same maturity level. This could result in less confusion to the reviewer and reduction of unnecessary contractor comment adjudication with few products under construction subsequent to artifacts being made available for review.</li> </ul>	<ul style="list-style-type: none"> <li>Requires more travel and resource allocation for review activities by PMO personnel</li> <li>Requires increased communications by technical staff, conveying interim review results to PMO management personnel.</li> <li>Impact: Additional communications required to ensure effective information flow from technical staff to PMO personnel. This communication must accurately present results and corresponding context for each iterative review. Further, non-verbal aspects of the iterative review as well as management level nuances can be difficult to capture in prose.</li> </ul>

Precedents exist for each of these options being used in DoD acquisition programs, in some cases dating back to the 1990s. The key points to making these options effective are

- open, honest, and frequent communications
- careful review and tailoring of the milestone event criteria (entry/exit and expected results)
- comfort being outside the traditional acquisition life cycle “comfort zone”

The next section of this document addresses the execution of technical reviews in iterative/agile environments.

---

### 3 Performing Effective Technical Evaluation in the Iterative/Agile Environment

This section explores what it means to do technical evaluation in the context of the iterative or agile environment in comparison to how it would be done in a traditional acquisition environment. In the context of iterative software engineering processes employed by some programs, many of the tasks are the same, but their focus will be slightly different. We discuss the roster of participants required to carry out technical evaluations and the adaptations required of those participants before turning toward specific points about technical effort estimation and program oversight.

#### 3.1 Technical Evaluation Participants

Technical evaluation teams should include representatives from the teams/disciplines that comprise the program office. In our experience working with a variety of programs across the DoD, participants should include (but are not limited to) representatives from the following disciplines:

- program management
- systems engineering
- software engineering
- information assurance/security
- hardware/manufacturing
- logistics
- contracting
- financial management
- test and verification

This list of representatives is typical of the historical representation found during most traditional technical evaluations. The type and domain of the participants does not change for iterative or agile environments.

#### 3.2 Cultural Adaptation to Iterative Methods

Cultures often emerge around the methods that are employed on a project. This section addresses the iterative principles that can be used by contractors and how they compare to traditional principles. This should help the government understand the differences and define processes to enable changes that accommodate these differences.

The traditional culture that most government PMOs are familiar with is different from the one that is emerging out of iterative or agile developed programs. Neither is inherently better than the other, but one may be more suitable than the other for different purposes. Until alternative direction is available, the challenge is to facilitate the adoption of the iterative culture within the constructs of the traditional DoD culture. Steps will need to be taken to mitigate risks related to cultural conflicts that may arise. Table 3 provides an overview of the cultural dimensions that may be affected [Lapham 2011].

Table 3: Comparison of Iterative with Traditional DoD Cultural Elements

	Iterative DoD	Traditional DoD
<b>Organizational Structure</b>	Flexible and adaptive structures Strong communication mechanisms when teams are distributed	Formal structures that are difficult to change Hierarchical, command and control-based teams Integrated Product Teams that have formal responsibilities
<b>Leadership style</b>	Facilitative leadership Leader as champion and team advocate	Leader as keeper of vision Leader as primary source of authority to act
<b>Rewards System</b>	Team is focus of reward systems Sometimes team itself recognizes individuals	Individual is focus of the reward system
<b>Communications and Decision Making</b>	Frequent interim reviews such as PCDW, PIRR, PPDW, PBPR, and PSDW) Evocative documents to feed conversation "Just enough" documentation, highly dependent on product context	Top down communication structures dominate External regulations, policies and procedures drive the focus of work. Indirect communications, like documented activities and processes dominate over face-to-face dialogue Extensive traditional, representational documents used by the PMO throughout the development life cycle to oversee the progress of the developer PMO oversight tools focused on demonstrating compliance vs. achieving insight into progress
<b>Staffing Model</b>	Cross-functional teams including all roles across the life cycle throughout the lifespan of the project	Uses traditional life cycle model with separate teams, particularly for development and testing Different roles are active at different defined points in the life cycle and are not substantively involved except at those times

In order to adapt to the newer iterative culture, adjustments in expectations will have to occur. Some of the adjustments that should be considered are

- providing incentives based on early incremental software release as opposed to "finished" documents, thus planning those incremental releases and incorporating them into the award/incentive structure on the contract. This will most likely change the type of reward system in place.
- participating actively in progress reviews and demonstrations (incremental or progressive reviews)
- planning and working collaboratively (shoulder-to-shoulder) amongst the program team, including both customers and management (acquirers and contractors)
- actively partnering in the creation of the software solution. This would include collaboration during RFCS.

The GAO has repeatedly testified before Congress that it recommends that program managers encourage collaboration and communication.<sup>7</sup> Both developer and PMO need to reflect a “learn and adapt” viewpoint rather than a “big bang” viewpoint. The PMO should communicate frequently with the contractor about program goals, provide timely feedback on iterative software capabilities as they are developed, and work with the contractor to ensure changes are well understood and smoothly incorporated as required and approved. The contractor is responsible for developing the software but uses the synergy gained from regular interaction with the PMO to meet expectations and refine the capabilities being developed. Overall, this synergy and constant communication should help reduce programmatic risks. Note that “collaboration” is not meant to encourage constructive change, but rather to allow for detailed discussion of the change in question.<sup>8</sup> If other changes are identified, then they must be entered into the RFC process for approval.

The following sections address specific topics that generated questions when we were working with a large program using iterative development.

### 3.3 Collaborative Environment

Agile or iterative software developments are based on collaboration between the contractor and customer. Without direct involvement of the system’s intended users or knowledgeable surrogates (PMO members), early validation of the direction of the implementation is impossible to accomplish. Early validation of the direction is essential to ensure that iterative development proceeds in a stable manner, based on layer after layer of evolving, useful functionality. Some programs may refer to these evolving layers as “threads,” “releases,” “builds,” “iterations,” “spirals,” “slices,” and other similar terms.

The DoD recognizes the benefits of incremental approaches to system development. The Defense Acquisition Guidebook (DAG) Section 4.3.6 discusses the use of iterative development methodologies in the context of incremental acquisitions [DAU 2011] (as previously noted in 2.1.1):

*In incremental development, capability is developed and fielded in increments with each successive increment building upon earlier increments to achieve an overall capability. These approaches to evolutionary acquisition are particularly effective in quickly fielding an initial capability or increment of functionality while allowing continued efforts to incrementally attain the final, full, end-state capability.*

A contractor’s iterative or agile methodology should employ similar concepts. Specific behaviors emerge when contractors employ iterative methods. These behaviors will require certain aspects of the traditional activities of both the contractor and the PMO to change. Programmatic oversight objectives do not change; however, the methods for achieving these objectives will vary from

---

<sup>7</sup> GAO-10-447T (2010), GAO-11-590T (2011), GAO-09-705T (2009)

<sup>8</sup> Constructive change is “defined as an oral or written act or failure to act by authorized Government official construed by contractor as having same effect as a written change order. Such a change must involve (a) a change in performance beyond minimum contract requirements, and (b) word or deed by government representative which requires contractor effort that is not a necessary part of the contract, and (c) it requires ratification.” [Lap-ham 2010]

those undertaken in more traditional waterfall-based developments. Some of the government PMO enabling actions that support iterative methods are the following:

1. Adjust expectations for document reviews, program milestone reviews, and progressive reviews to recognize that work products will mature at varying rates in an iterative development. Requirements and design allocated to future iterations should not be expected to be fully matured during early iterations. Ensure that program office oversight is consistent with the expected maturity of the work products. Costs for these activities should reflect this approach. This avoids rework by allowing for learning from earlier iterations to gracefully inform the later parts of the design and implementation in conjunction with the evolution of the capabilities to be developed.
2. Allocate resources and emphasize to program office staff and other stakeholders the importance of participating in the joint progressive reviews such as such as the Progressive Critical Design Walkthrough (PCDW), PIRR, PPDW, PBPR, and PSDW *in addition to* the programmatic milestone capstone events (PDR, CDR, etc.). These progressive reviews are used to build common understanding of incremental capabilities, layer after layer, which will form the overall capability seen at the large program milestone reviews and are literally where program guidance decisions are made. Thus, the development/government feedback at the reviews should be as important, if not more so, than that from the program milestone reviews. Diligence in support of these activities will improve communication between the program and the contractor and save effort and cost further downstream. Cost estimates should reflect this type of government/contractor interaction.
3. Employ candor, open communications (including face-to-face meetings as appropriate), and transparency, in particular during the negotiations of RFCs. This includes ensuring prompt turnaround of CDRL approvals.
4. Ensure developer access to end user surrogates at the PMO with depth in operational use of the proposed system capabilities. The PMO must be sure that user/operational requirements are accurately translated to the contractor to ensure clarity and consistency, and that any contractor questions about the requirements can be resolved expeditiously and accurately. This resolution will reduce the overall cost associated with any RFC.
5. Establish CDRLs and other contractual criteria that encourage several small deliveries rather than a single “big bang” delivery. In the DIDs for these CDRLs, ensure that the correct expectations are set in terms of the maturity of the information being commensurate with the maturity of the release being worked on.
6. Permit early delivery of working (although potentially prototype-quality) software. This allows misunderstandings to be caught early and provides both developer and user the opportunity to refresh their knowledge on the evolving operational challenges that the software will be expected to meet.
7. Ensure that both the contractor and the government PMO agree on what “done” means. Remember that work products will necessarily exist at varying levels of maturity due to the iterative nature of the approach. Interim work products must meet pre-established completion criteria to move into a final, releasable state.
8. Identify and employ someone on the government side to coach/train program office staff in understanding and working within the iterative environment. While most program offices are not staffed to “coach” methodology, they are funded or have access to training for their

teams. When embarking on programs that use or plan to use iterative or agile methods, training on those methods is paramount, just as training on the acquisition process in general is required. The objective is to ensure personnel involved in evaluation of cost estimates based on iterative/agile methods know what differences to expect and how to identify them. Training and coaching activities can run the gamut from formal professional courses/seminars, to training days, to brown-bag lunches with those experienced in these methods.

The key behaviors above are ones that will help the program move toward allowing the contractor to provide benefits from employing the iterative development methodology, while still living within the traditional DoD acquisition structure.

The most important of any of the enablers is to ensure that there is strong cooperation between the PMO and the contractor. This is critical to realizing the benefits of an iterative approach. This cooperation will facilitate obtaining an independent cost estimate in that more information and data should flow (in both directions) during the collaboration. Assumptions, past performance, and other key points should be shared. This will build understanding, ensure the requirements are interpreted correctly, and reduce risk. By cooperation, we mean communication about ground rules and assumptions and other general factors that contribute to shared understanding. For example, one RFC<sup>9</sup> we were told about had estimates from both the PMO and the contractor. These estimates were 400 and 1,000 SLOC respectively. A collaborative discussion was held to address the issue, resulting in new estimates from the contractor, new government negotiation limits, and an agreed-upon resolution.

The notion of collaboration as a key to successful program execution is not a new one and is not exclusive to iterative or agile development. However, such collaboration is an absolute requirement to realizing the benefits of iterative approaches. Collaborative behavior might actually be *easier* to accomplish in an iterative setting because the scope of an iteration is limited—it is considerably smaller than the entire program scope.

### **3.4 Common Areas of Contention: Systems Engineering and Test**

Any requested change to the system ripples into a reassessment of the systems and test engineering efforts and resources required to execute the program. In our experience with programs leveraging agile methods, the application of systems engineering and test engineering effort to RFC estimates has been an area that causes a great deal of consternation and miscommunication between contractors and government program office teams. The methods that contractors have used to arrive at modified estimates for these tasks are likely not as straightforward or transparent as program office teams expect based on experience with more traditional development methods. As stated previously, this technical note is based on experiences dealing with technical proposals responding to RFCs to existing contracts. While many if not all of these observations may apply to technical evaluations from initial acquisition through sustainment, investigation of this claim remains for future research.

This section provides guidance for program office interpretation and evaluation of contractor estimates for two critical areas: systems engineering and test.

---

<sup>9</sup> The authors were allowed to observe and review some RFCs from a program using iterative/agile methods. The program has requested anonymity.

### 3.4.1 Evaluating Systems Engineering Estimates

Typically, for traditional systems comprised of subsystems or segments, the subsystem system engineering hours are determined by applying some factor against the development hours. Sometimes the contractor may consider the subsystem and segment system engineering hours as outside their typical iteration process and thus apply a factor to their development hours to obtain the systems engineering hours.

There is nothing incorrect about this approach. However, the basis for the development hours needs to be clearly disclosed. In the case of agile/iterative methods, initial estimation is likely to be performed using story points<sup>10</sup> or some other unit of work that is unique to the particular development team and defined by the contractor. If systems engineering effort is to be added to the estimate, some adjustment factor or estimating relationship must be applied that clearly links the estimated development effort to the add-on systems engineering effort. For a complete understanding, the PMO needs to obtain all the assumptions made in determining this formula. The following should be considered:

- If the RFC technical proposal affects work already done in a previous iteration, expect some rework in systems engineering. The contractor needs to explicitly state what work has already been done and what changes will be needed.
- If the RFC technical proposal affects work not yet done in any iteration and does not significantly change the scope of the feature or capability, in general, the system engineering hours shouldn't increase. However, if the scope is significantly changed, then additional or potentially fewer system engineering hours may be applicable. Again, all the technical and process assumptions related to the RFC need to be shared and discussed.

### 3.4.2 Evaluating Test Engineering Estimates

RFC technical proposals also necessitate an evaluation of the effort required to develop and execute all pertinent test cases. As with system engineering, test engineering hours for traditional systems are determined by applying some factor against the development hours. Note that subsystem and segment (if applicable) test engineering hours are *not* those hours expended during unit test or initial integration during the iterations.

In general, as with systems engineering, the following should be considered:

- If the RFC technical proposal affects work already done in a previous iteration, expect some additional testing. The contractor needs to explicitly state what work has already been done and what changes will be needed.
- If the RFC technical proposal affects work not yet done in any iteration and does not significantly change the scope of the feature or capability generally speaking, the test engineering hours shouldn't increase. However, if the scope is significantly changed then additional or

---

<sup>10</sup> "Story points are a unit of measure for expressing the overall size of a user story, feature, or other piece of work ...The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story. Rather a story-point estimate is an amalgamation of the amount of effort involved in developing the feature, the complexity of developing it, the risk inherent in it and so on." [Cohn 2006]

fewer test hours may be applicable. Again, all the assumptions need to be shared and discussed.

- RFC technical proposals may, in fact, result in a reduction in the scope of testing when functionality is cut from the system. This may not produce linear cuts in test cost/effort.
- Validation and verification in an iterative collaboration are reversed. Validation of requirements and implementation approach occur during each iteration through the user collaboration. Verification occurs via continuous integration and frequent regression testing of the evolving software code base. Careful review needs to be done to assure the proposed solution will be verified at the right level. For example, if a change is being proposed after the original code is verified, additional retesting will result. A requirement might be traded away at one iteration but then revised in a subsequent iteration. Ensuring the feature is tested in a subsequent segment verification run is then crucial.

A common practice of test engineering in agile or iterative development approaches is to engage in Test-Driven Development, where software code isn't written until the tests that will verify the code have been completed. In programs that are adopting this approach, test engineers are involved in a significant way from the very beginning of the project, and see the evolution of the software in a direct way.

### **3.4.3 Using Actual Data as a Basis of Estimate**

With both systems engineering and test engineering activities, past efforts on the program may be the most viable source of estimates in support of the RFC technical proposal. To that end, consider the following questions about the viability of actual program data:

- When using actual hours as the basis for the RFC technical proposal estimate (while this is a valid indicator of what it has taken to do the work), several questions need to be asked (these are not in any particular order):
  - Is the future work similar enough to the past work to warrant direct comparison?
  - Are there any circumstances (learning curve, changes in personnel) that may affect the estimate going forward? Agile environments are particularly sensitive to personnel change, since the relative estimation that is used inside an iteration is based largely on the particular skills and knowledge of the team performing the work.
  - Has the team reached a point where it is working at optimal levels, thus perhaps being able to reduce the amount of work needed to accomplish any new tasks? Optimal does not mean the team cannot continue to improve but in this case means that any learning curve and other contributing factors to forming the team are in the past.
  - How is the work tracked for each iteration? Are measurements and plans (and divergence from plans) being updated in a timely fashion to reflect the actual work of each iteration?
- Be aware that estimates for new/updated taskings may unintentionally mask problems or deficits in the current work. Understand how each variable in an estimation formula is calculated.

- For example, a formula that includes Estimate To Complete (ETC) for existing work in the calculation of a new effort estimate would inherently absorb any delays or rework caused by defects or errors that had previously been introduced in the existing work.
- To what extent is continuous integration and automated testing used and how is it impacted by the technical proposal?

In many of the agile development methodologies work is purposely delayed or rework (refactoring) is purposely planned to later iterations. These questions and their answers will help provide the flexibility that the PMO is expecting by using the iterative or agile process.

### 3.5 Oversight and Insight Implications

Every program requires oversight and insight by the government and contractor's program management offices. The required level of oversight is pre-determined by myriad government regulations. Assuming the program is designated as ACAT I, it requires significant oversight as outlined in the FAR and DoD 5000. Reviews such as PDR and CDR are required and will be part of the contract. However, the documentation typically created in traditional developments with these large milestone events is not separately created when using an iterative approach. The information is often available from the standard tools that are used to support iterative development, but creating formal documents only for review purposes is not typical of agile/iterative methods. Due to the iterative process, project documents and other CDRs will exhibit varying levels of maturity at the time of programmatic milestones. Difficulties may arise when evaluating changes that affect components at different levels of maturity. The key to avoiding these types of issues is to determine what is sufficient for the situation and implement the appropriate level of detail.

The entire purpose of technical assessments is to measure technical progress and assess both program plans and requirements. A structured technical review should demonstrate and confirm completion (within cost and schedule targets) of required technical accomplishments and exit criteria as defined by the program [DAU 2011]. There is a direct relationship between the technical review and software estimation processes. The output of the technical reviews becomes input for estimates for future iterations.

Some of the main challenges to the adoption of iterative methods include the following:

- lack of team-based incentives that promote cooperative interactions and communications while remaining within the bounds of the FAR and other government regulations [Lapham 2011]
- difficulty in establishing incentive structures that reward both working incremental software delivery and sufficient (at appropriate maturity) documentation [Lapham, 2011]
- lack of shared understanding of definitions/key concepts (Consider SLOC-based notions of program size vs. story points)
- lack of understanding of document content and volume. As previously noted, documentation in agile development efforts matures through the course of development. When program office teams fail to recognize that evolution, this can mire down insight and oversight activities.

- daunting regulatory language—Acquirers may fear agile efforts because it is currently less straightforward to establish and maintain traceability between applicable regulations and the evolving documents that are produced in an agile delivery [Lapham 2011].

All of these issues will, to some extent, impact the estimation process and the related negotiations.

Most things that occur during contract negotiations (including those for RFCs) using iterative methods appear to be very similar to traditional negotiations; however, differences are present. For instance, a contractor may not be able to use actual program data to forecast systems engineering effort associated with an RFC due to a lack of available data on the program to date. Data may be unavailable or unreliable simply because it is early in the program, only one or two iterations have been executed, rendering the sample size too small to use as a basis to determine the amount of overall systems engineering required to support the change. Another example is when the contractor says it is using actual hours for some time period. In agile terms, it is using its teams' velocities for that period of time. While the contractor should be providing the translation to traditional estimates, the government could lose insight into the lowest levels of the basis of estimation. It is critical that the government team understand the linkages and traceability associated with such translations. The contractor's SEMP, measurement plans, and other program documentation should paint a clear picture that enables the government team to follow the derivation of cost and effort estimates.

Furthermore, maturity of deliverables and the data provided must be understood *in advance*. Agile methods are not an excuse for, nor are they synonymous with, poor program tracking. The government team must have clarity about the expected evolution of tracking data and monitor status reports and estimates accordingly.

Finally, the government PMO must understand that it has contracted for an iterative development method that is being implemented within a traditional DoD acquisition model. This distinction was described in Section 2 of this document.

The next section of this document addresses the actual review of a contractor's estimation methodology in an agile development/DoD acquisition model.

---

## 4 Assessment of Contractor Estimating Methodology

This section will address the types of data the program office should expect to receive from the contractor about their estimation methodology. In addition to any estimating and basis of estimate documentation, other supporting documents including, but not limited to, the Software Development Plan and the Master Software Build Plan also provide insight into the methods employed by the contractor for estimating their work.

Table 4 in Appendix B provides a distillation of key questions government personnel might ask regarding contractor estimation methodology. It may be used to guide further critical analysis on the part of program office staff in support of RFCs that involve iterative or agile development methodologies. The sections included here focus on different types of information that should be documented and understood by the evaluation team.

### 4.1 Methodology Documentation Review

Absolute estimation methods, typical of those used in evaluating RFC technical proposals, require definition of the “unit of work” that will be used as the basis for allocating resources and schedule. In traditional software development methodologies, function points (FPs) or source lines of code (SLOC) have been the most prevalent definitions for a unit of work (UOW). As noted in 3.4.1, in agile/iterative methods, initial estimation is likely to be performed using story points or some other *relative* unit of work that is unique to the particular development team and defined by the contractor.

Upon initial review, the methodology documentation provided by some agile developers may leave the reader with the impression that estimation for software elements is done on the basis of source lines of code (SLOC)-based sizing perhaps through the use of parametric models. Often this is done because the developer believes that the traditional sizing is how they are expected to communicate their estimates, even though the SLOC estimate has actually been derived from another UOW type.<sup>11</sup> However, a review of the Software Development Plan (SDP) is likely to show that a different UOW such as a story point (or something similarly defined) is the building block for assigning technical taskings across an iteration. The methodology documentation may fail to establish the linkage between these bases of estimation.

For the evaluator to truly understand and have confidence in the viability of the estimates, if such a conversion is made between UOW and SLOC counts, it must be documented and made available to the government team. Questions the team should ask when reviewing the conversion information include the following:

- Is the fidelity of the estimate the same level before and after the conversion?

---

<sup>11</sup> In the programs observed for this technical note, this type of conversion was done to allow the government to see estimates in familiar terms. In fact, this instance did use parametric tools while not all do. This is not a normal agile practice but one adopted in this instance. In cases where both the contractor and the government are using agile specific models added to parametric tools, this conversion to SLOC may not occur.

- Is there a clear linkage point with conversion data that allows the estimation process to move seamlessly from a UOW-based estimate of engineering work to perform, toward a SEER-SEM-based (or other parametric tool) estimate of the size of the resultant work product?
- If a parametric model is used directly, does it employ tool components or modules specific to agile methods? If not, have factors been adjusted *a priori* to account for differences attributable to agile or other iterative methods?

Furthermore, review of the SDP during a RFC technical evaluation may show that the “size” definition of a UOW has changed since the original SDP was submitted;<sup>12</sup> if so, such a change invites important questions:

- Why has the effort range for UOWs shifted?
  1. What is the rationale?
    1. Are data available to support the change?
    2. Is the change a result of poor performance or performance variations?
    3. How has the contractor determined that a different size range better encapsulates the individual tasks? (this is actually quite likely, since the contractor will have higher fidelity team velocity data once iterations have actually commenced)
  2. What is the impact of the change in scale?
    1. Is the sizing of the UOW merely an adjustment of scale for staff management purposes, or do downstream calculations and estimations require revision?
    2. Does the quantity of UOWs remain constant, with a corresponding overall schedule increase?
    3. Or does the number of UOWs decrease, while overall effort/schedule planning remains constant?

#### 4.1.1 Work Breakdown Structure and Skill Mix

A short discussion providing background for how program-specific factors are developed is helpful for understanding why certain metrics are used and others are not in different estimation packages. If the Work Breakdown Structure (WBS) and skill mix are provided, a discussion of how the actual data is distributed across the WBS and skill mix would be helpful. This explanation should include the factors used to create any estimates and how they were derived, as well as the labor or cost implication for each affected line of the WBS. Also look for any dates or revision numbers on the WBS and skill mix data. This determines how current the data is and if it has changed multiple times. If it has changed, an explanation of why it has changed would be germane. While detailed information about skill mixes used within an agile team were not gathered

---

<sup>12</sup> Many might think the SDP should define how the UOW is supposed to change based on past estimations and actual velocity obtained. In this case, the contractor did not put that information in the SDP but rather just stated that a change occurred. This behavior invited the questions in the list. Even if a change is defined, the questions are pertinent and should be addressed during the review of the SDP.

during this study, note that the skill mix and make up of an agile team can be different than that of a team using traditional software methods.

#### **4.1.2 Estimating Methodology**

A good top level overview of the contractor's estimating methodology should be provided. Look for several details that provide greater insight such as information about the estimate's assumptions, the basis of estimate, parametric models and how they were used. The government program office needs to totally understand the contractor's estimation methodology so they can develop a way of translating it to something that will be acceptable to upper levels of government. Sometimes the contractor will provide this type of data by providing the information in the standard format versus the format used by the agile team. If the translation is needed, then it could include a training process to educate everyone in the status chain on what the contractor's data means. Each of the topics is addressed in the following subsections.

##### **4.1.2.1 General**

These high level provisions apply to all elements of the cost/effort calculations in support of a proposed change.

1. Make sure any calculations provided are accurate. Inappropriate/unclear mathematical representation hinders comprehension of the data and causes concerns about the fidelity of the data in other reports/deliverables. While this is true for both traditional and agile estimations, it can become more onerous in an agile environment as people are learning that method and may overlook the basics.
2. Look for an explanation of how iteration tasking and staffing are arranged to minimize conflict among staff resources and maximize staff skill sets. If this data is not available, then ask for a short explanation. While agile enables self-directed teams, this is not an excuse for not having this data available.
3. Overlap across iterations helps to minimize schedule but increases dependencies and the probability of rework. Assumptions regarding iteration overlap should be explicitly stated in the basis of estimate.
4. Probe whether drawdown from the project backlog reflects risk reduction, versus a deferral of difficult work until later iterations. Ask specific questions about why the different items in the project backlog are chosen for specific iterations.

##### **4.1.2.2 Basis of Estimate**

This section addresses the validity of core data and assumptions used to derive estimates in support of the RFC.

- Determine how labor hours are estimated. Are they based on historical experience or experience on similar (relevant and analogous) previous programs, modified to reflect the complexity in program specifications and requirements, manufacturing processes, integration procedures, make-buy decisions, etc.? Ensure the following questions are answered:
  1. Is historical information about agile development metrics available on the actual work team proposed to handle the task? Did the historical project use the same agile processes?

2. What did the contractor use if historical or similar previous experience is not used?
3. If theoretical data is used, how will it be adjusted to factor in historical performance as the project progresses?
4. How does contractor handle extraordinary circumstances with historical experience such as excessive rework or refactoring? How does the contractor determine similarity?
5. What similar programs are considered as a basis for historical estimates? Or will this vary greatly based on the requirement/RFC?
6. Are the basic components of the work at hand the same as the UOW that is defined in the SDP? If “basic work components” are not UOWs, the linkage between all of these sizing elements should be explained in further detail.

- Ensure the contractor provides the method for identifying or determining analogous or equivalent efforts if they use that data to spread software efforts across the scheduled time frame. This information is critical for the government to understand the methodology.
- Many times the calculations for systems engineering and test engineering are based on actual to date and perhaps Estimate to Completion (ETC). These numbers need to be compared to historical data. Variations need to be identified and justified. Other questions on this topic include the following:<sup>13</sup>
  1. Is there sufficient analysis to justify why the actual effort to date is higher or lower than anticipated?
  2. How do any adjustment factors (such as effort variance factors and cost estimating relationship) on this project compare to those observed on the other analogous/similar programs which have been identified by the contractor and used to aid in the preparation and validation of estimates on this program?
  3. Is there sufficient analysis to compare the complexity and overall tasking of future work to that already performed to justify the basis for estimate/comparison?
  4. Check to see that all source data (such as the WBS) are clearly marked with an “as of” date. This simple information also facilitates better historical analysis by the government team in the development of independent cost estimates for future RFCS.
  5. Is the continuous integration and test implementation on schedule? Is the cost as planned? Is the automated test suite working as expected and being continuously improved?
  6. Is the challenge of testing being taken into account as far as the overall iteration schedule is concerned?

---

<sup>13</sup> Note that many of these questions are the same as one would ask for any traditional way of doing software estimates. In fact, that's the point. The “what” you do is still the same, “how” you do it is different. The “how” the contractor does estimation in their agile environment is what the program office needs to understand.

7. How does the contractor account for extra integration effort required to keep any reused software compatible to evolving versions of the program software?

#### **4.1.2.3 Parametric Models**

Parametric models (such as COCOMO, SEER-SEM) have long been used to aid in software engineering estimation. This section addresses the use of such models by contractors engaged in agile methods, to enable the program office to gain understanding of the fidelity of estimates derived using one or more of these models.<sup>14</sup>

- What, if any, parametric method was employed in the estimation process? Are available Agile extensions to the basic models being used? What factors were employed, if any, to adjust estimates for variations in scope or other unique project attributes?
- How was the software productivity projected by parametric data (i.e., the outputs of parametric models) compared to the average productivities experienced by their team members on similar programs:
  1. What characteristics are used to determine similar programs?
  2. What are these relevant and analogous programs?
- Were the outputs to average productivities experienced by their teams compared to the parametric predictions?

#### **4.1.3 Additional Resources**

Many times data needed for estimating a RFC technical proposal is found in documents other than the costing documents. The government program office personnel need to be familiar with any data found in the SDP, MSBP, and SEMP that impacts estimation.

##### **4.1.3.1 Software Development Plan**

The SDP, whether iterative or not, provides ample information for a new software engineer on the program to understand the work to be performed, how it is to be performed, who is responsible, etc. It should make appropriate use of cross-referenced documentation (e.g., SEMP, MSBP, and the like) without repeating information. Some specific topics that are useful to include, when agile or iterative methods are in use, include the following:

- How does the agile process deal with changing requirements?
- What aspects of the process will be automated, and how will this be accomplished?
- Explain how the contractor plans on aligning the traditional acquisition milestones (e.g., PDR, CDR, etc.), if used, with the increment and/or iteration milestones.
- If Earned Value (EV) is used, identify how the EV is applied for the iterations and/or increments (e.g., 0/100, 50/50, etc.) as well as what tasks are considered discrete tasks vs. level-

---

<sup>14</sup> There are many references about using parametric models for agile projects. For a high level introduction, see Agile Methods: Selected DoD Management and Acquisition Concerns, <http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=9769>

of-effort (LOE) tasks. All parties must agree on what “done” means! The government needs to be involved in the planning of each iteration similar to rolling waves.

- Determine if the UOW definition used for the initial costing is different than that used for RFC estimation.
- For ACAT I programs, make sure a Software Resources Data Report (SRDR) is included if the SRDR provides software-related data to the government to improve the ability of the Department of Defense to estimate the costs of software intensive programs. Data collected from applicable projects describe the type and size of the software development, and the schedule and labor resources needed for the development.
- The treatment of COTS, GOTS, NDI, and other reused software must be included in the SDP. Consider: how will these products and their integration be tested? When will they be integrated? How does the iteration or agile process account for COTS, GOTS, and NDI?
- If iterations are used, the progressive reviews should follow essentially a similar process you would find for a PDW, CDW, ITR, and the like, but focused only on the potentially deliverable software for that iteration. Large programs will sometimes have a PDW followed by a PDR, CDW followed by a CDR, etc. Where that particular system is utilized, the PDW/CDW will be a very technical meeting (sometimes lasting a week depending on the scope<sup>15</sup>) with each PDW/CDW providing part of the technical piece of the final PDR/CDR. The PDR/CDR would be more of a PM-level review that includes a summarization of the program to date as well as cost, schedule and any items from the prior “walkthrough” event that needed to be briefed.

#### 4.1.3.2 Master Software Build Plan

The Master Software Build Plan identifies the software development activities, artifacts, and Independent Verification and Validation (IV&V) threads. It provides some amount of detail with respect to the software build planning approach and how the various functions were mapped to the blocks and subsequent iterations. The MSBP should do the following:

- demonstrate the use of continuous integration, automated check-in, and test automation as part of the planning and implementation of the iterative or agile method
- provide plans for defining how software requirements and functionality are allocated to development phases (blocks and iterations within blocks), thus defining the overall phased approach to building capabilities incrementally
- offer a detailed explanation of how the Master Software Build Planning process is performed along with the participants, entry criteria, inputs, tasks, tools and methodologies, software measures, outputs and exit criteria, and the definition of done
- summarize the life-cycle model (iterative-incremental life cycle) being used and how it relates to the rest of the MSBP. It should also lay out the nominal “x” week iteration cycle (where x is the number of weeks within each iteration).
- describe high-level block and iteration activities (within each incremental block). This information is similar to what you would see in any typical software development life cycle

---

<sup>15</sup> The program reviewed for this technical note was large and averaged a week per these types of reviews. Depending on the size of the iteration and the amount of software, the review could be shorter.

(preliminary design, build planning, detailed design, code and unit test, unit integration and test, software integration and test). Depending on the chosen agile method, the makeup of the iteration activities will be some variant of this list.

- illustrate how the contractor divides the iteration across the various engineering activities as well as the percentage of effort required by each engineering activity/discipline. This information should pretty closely match that found in the contractor's Control Account Manager (CAM) book for the work activity (e.g., efforts, duration, costs). Keep in mind this type of information is not usually something an agile team would deal with. Thus, negotiations on the exact data including definitions of content may be required.
- ensure the government or its authorized representative has access for review of software products and activities, from both the prime contractor and its subcontractors, in accordance with the Statement of Work (SOW) and upon approval from program management. This access should allow the PMO to attend the iteration and progressive walkthroughs, etc. to gain insight into the current development status and progress of the program elements.
- describe the metrics used to provide insight into the program (including Software Earned Value [EVMS] if applicable) and the use of Quantifiable Backup Data (QBD)<sup>16</sup> for reporting.

#### **4.1.3.3 Systems Engineering Master Plan**

The Systems Engineering Management Plan should provide an understanding of how the progressive reviews feed the programmatic reviews and vice versa. The SEMP should include the following:

- clear definition and description of progressive technical reviews and audits. These should be designed and implemented to minimize the risks associated with using this technique. Program impacts of progressive technical reviews and audits must also be addressed.
- definitions of mechanisms and metrics for monitoring and controlling any movement of requirements between/across iterations
- a description of how the Discrepancy Reports (DRs) feed back into each iteration/block being developed and the baseline controls should be provided
- detailed and straightforward definitions and descriptions of formal program-level reviews found in any acquisition (SDR, PDR, CDR, etc.) and how they may be different in the iterative environment
- a discussion of Joint Technical Reviews, the various reviews and the form(s) they may take. As these are considered “joint reviews,” the government should be participating in them the same as any other milestone review

---

<sup>16</sup> QBD is referred to as “a detailed listing of tasks necessary to complete all scope in a work package during the defined period of performance.” The package features weighting on each element. [Allerman 2009]

---

## 5 Conclusion

Agile and iterative methods are not orthogonal to the DoD acquisition life cycle. However, government and contractor teams must work together closely to ensure success when these methods are employed on a DoD program.

Program offices need to be aware that there are differences in how to evaluate the proposed work and subsequent change estimates that are created using iterative methods versus those developed via traditional methods. While all the same types of activities still occur, they are accomplished in a slightly different manner—notably, work products will exhibit varying levels of maturity throughout the program life cycle, setting expectations for the agile or iterative environment and process, changing the mode of communications to complement the agile method, and identifying the specific milestones for measuring progress and accepting the system. If one uniformly applies traditional review criteria to an iterative estimate, a great many disconnects and issues will occur. Program office personnel need to take their current knowledge and use an “iterative-focused lens” to interpret RFC responses. The authors have provided resources to help bridge the gap between understanding traditional and iterative projects and provide a list of pertinent questions to answer when doing a review of iterative technical proposals to a RFC.

While these same challenges may apply anytime during the life cycle of the program when technical evaluations occur, from pre-award through sustainment, no data was captured for other than RFCs. This research will be addressed at a future date.

---

## Appendix A Evaluation Question Checklist

For convenience, clarity and ease of reference, the questions raised in this technical note are repeated here in Table 4. A word of caution: When using these questions, make sure you are applying them with an iterative model in mind, since applying them while thinking traditional methods will not result in the same answers. The Consideration column provides some additional guidance on interpreting answers that may be offered in response to the questions.

*Table 4: Questions to Ask When Evaluating Iterative Proposals*

Question/Action	Consideration / Impact
To understand and have confidence in the viability of the estimates, the conversion between UOW and SLOC counts must be documented and made available to the government team. Note that this is not the normal course of action for an agile project but given the significant difference from business as usual some accommodation needs to be made. Ensure the definition of the terms do not change when document versions change. If so, find out why.	Consider if the fidelity of the estimate is the same level before and after the conversion. If not, adjustments need to be made to preserve the fidelity. Look for a documented process that moves from the UOW estimate to a SEER-SEM-like input. This process would help in eliminating any fidelity issues caused by the conversion. It would also put the estimates into a more familiar representation for the government personnel.
Ensure the effort range for UOWs is the same as previous definitions.	If the effort range (number of days) for the UOWs shifts then an explanation needs to be provided. Enquire if the shift is merely an adjustment scale for staff management or if other downstream calculations require revision. If the UOWs change, what is the impact to the overall schedule? This could mean either more or less work is being accomplished within the same timeframe (which could mean a schedule increase or decrease).
Inquire how program-specific factors were developed. What method is used for estimation – story points, planning poker, user story counts – make sure it is clear how the contractor arrives at their estimation. If prior experience is used as a factor, determine how analogous that program is to yours and if the methods used were the same.	This would further help the government understand the methodology and the details behind the basis of estimate.
Ask how the actual data is distributed across the WBS to include the factors used to create any estimates and how they were derived. Determine if the WBS is being changed because of the response to the RFC.	This would further help the government understand the methodology and know what WBS elements are changed due to the new estimates.
What is used if historical experience or similar to previous programs estimating data is not available?	Typically, the contractor will use historical experience on similar programs to provide a foundation for the estimates. If similar programs are not available, the government needs to understand what is being used as the foundation for the estimate.
How does the contractor determine similarity of other programs for purposes of comparison/historical data? Do they take into account the instantiation of the agile method and any variations specific to the other program which may change the meaning of some items (done, milestones, etc.)	This data is critical if the estimate is to be considered valid.
How does the contractor handle extraordinary circumstances with historical experience such as excessive rework?	This will also help determine the validity of the estimate and what factors were or were not included.

Question/Action	Consideration / Impact
What similar programs are considered as a basis for historical estimates? Or will this vary greatly based on the requirement/RFC?	This helps determine the validity of the estimate.
Anytime a contractor states they are using adjustment factors (or a series of factors), ask what they are. Are any of these factors a result of using this particular instantiation of the agile or iterative method?	Specific data on what factors are being used will help the government understand the basis of estimate.
What characteristics are used to determine relevant and analogous programs?	This information allows the government to understand how relevant the other programs used really are to the program in question. It also serves to help determine the level of fidelity the estimates might have. For programs using agile or iterative methods, it is extremely important that the same processes and techniques are being employed on the current system, otherwise, errors can be introduced.
What are the relevant and analogous programs used by the contractor in the estimation process?	The answer to this question will allow the government to determine in their view if they agree that the programs are indeed relevant and analogous. If they are, then the estimation data is more appropriate to use. If they are not, then the government needs to discuss the topic with the contractor so that both parties agree on the scope of the tasking.
What does the contractor do if the assessment of SEER (or other parametric model) projected software productivity is analogous to other contractor experience shows that execution is not feasible?	This can point out an issue either with incorrect estimation or a lack of understanding of the problem to be solved.
If the contractor uses terms like "high degree," ask what constitutes a "high degree."	Terms like "high degree" are very subjective. A more definitive term needs to be employed or this term needs to be well defined. The definition should include a numeric definition of high degree such as 50 percent or 80 percent or 95 percent fulfillment.
Ask for a description of the process for accurately estimating the effort required to implement reuse software.	Answers to these questions will provide good background to understand the basis of estimate.
Ask for any guidelines or worksheets such as the contractor's software estimating guidelines and any supporting tools.	These will provide additional insight into the estimation process and allow the government to better judge the overall quality of the estimate.
How does the contractor assess "differences in scope" between reuse software and program requirements?	May provide more insight into that process might provide additional clarity to the RFC process.
Ask what guidance is available for engineers when using estimating ranges (for example, adjustment factors based on complexity) to ensure consistent usage.	The will help the government ensure that the estimation process is consistent across all estimators.
Ask for all formulas employed.	This would provide completeness and minimize the likelihood of misunderstanding.
When a similar program is cited, ask if the techniques used on both programs are the same/similar. If not, ask about notable differences.	This provides a more complete understanding for the government of the basis of estimate.
Is the contractor using automated testing? If so, to what extent?	This will help the government gain additional insight into the development of test estimates and the impact of a particular RFC on the test program. (Consider: does the change necessitate the redevelopment of automated tests? How does this impact the results of previous test runs, or the execution of tests on other functionality unrelated to the RFC?)
How does the contractor minimize conflict in resource utilization across iterations?	Careful and complete cross referencing will aid in the overall understanding of the estimation process.

Question/Action	Consideration / Impact
Do estimates for supporting RFCs erroneously double-count or unintentionally mask existing problems/deficits?	Verifying the variables used to develop estimates will improve the overall understanding of the estimation process and ensure that the outputs are valid.
When dealing with vendor quotes within an RFC, what criteria are used for realism, accuracy, and completeness?	Provide additional insight to the government.
Determine if the work to be done in the RFC is possible within one iteration or more. How will it be reviewed?	The entire idea of major milestone reviews needs to be included within any new costing. Are these reviews to be conducted relative to progressive reviews or at much larger system level reviews? The level of maturity within the documentation will vary depending on when it is reviewed.

---

## Appendix B Acronyms

BCL	Business Capability Life Cycle
CAM	Control Account Manager
CDD	Capability Development Document
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CDW	Critical Design Walkthrough
CMU	Carnegie Mellon University
COTS	Commercial-Off-The-Shelf
CPD	Capability Production Document
CWBS	Contractor Work Breakdown Structure
DAL	Data Accession List
DBSMC	Defense Business Systems Management Committee
DoD	Department of Defense
DRs	Discrepancy Reports
EMD	Engineering and Manufacturing Development
ERAM	Enterprise Risk Assessment Methodology
ETC	Estimate to Complete
EV	Earned Value
EVMS	Earned Value Management System
FRP	Full Rate Production
GOTS	Government Off-The-Shelf
PBPR	Progressive Build Planning Review
IBR	Integrated Baseline Review
PCDW	Progressive Critical Design Walkthrough
PIRR	Progressive Integration Readiness Review
IMS	Integrated Master Schedule
IOC	Initial Operating Capability
IOT&E	Initial Operational Test and Evaluation
PPDW	Progressive Preliminary Design Walkthrough
IRB	Investment Review Board
PSDW	Progressive Segment Design Walkthrough
IRR	Interim Readiness Review
PTRR	Progressive Test Readiness Review
IV&V	Independent Verification and Validation
JROC	Joint Requirements Oversight Committee
LRIP	Low Rate Initial Production
MAIS	Major Automated Information System
MDAP	Major Defense Acquisition Program
MDD	Materiel Development Decision
MSBP	Master Software Build Plan
NDI	Non-Developmental Item

PDR	Preliminary Design Review
PDW	Preliminary Design Walkthrough
PMO	Program Management Office
QBD	Quantifiable Backup Data
RFC	Request for Change
SBR	Segment Block Review
SDP	Software Development Plan
SDR	Software Design Review
SEER	System Evaluations and Estimation of Resources
SEI	Software Engineering Institute
SEMP	System Engineering Management Plan
SLOC	Source Lines of Code
SOW	Statement of Work
SPO	System Program Office
SRDR	Software Resources Data Report
TN	Technical Note
TRR	Test Readiness Review
UOW	Unit of Work
WBS	Work Breakdown Structure

---

## References/Bibliography

URLs are valid as of the publication date of this document.

### **[Allerman 2009]**

Allerman, Glen. *Deliverables Based Planning*, Denver PMI Symposium 2009.  
<http://www.slideshare.net/galleman/denver-pmi-symposium-2009>

### **[Cohn 2006]**

Cohn, Mike. *Agile Estimating and Planning*. Pearson Education Inc., 2006.

### **[DAU 2011]**

Defense Acquisition University. *Defense Acquisition Guidebook*, 2011.  
<https://dag.dau.mil/Pages/Default.aspx>

### **[DoD 2008]**

Department of Defense. DoDI 5000.02, *Operation of the Defense Acquisition System*, 2 December 2008. <http://www.dtic.mil/whs/directives/corres/pdf/500002p.pdf>

### **[DoD 2011a]**

Department of Defense. DTM 11-009, Directive-Type Memorandum (DTM) 11-009, *Acquisition Policy for Defense Business Systems (DBS)*, 23 June 2011.  
<https://dap.dau.mil/policy/Documents/2011/DTM%2011-009.pdf>

### **[DoD 2011b]**

Department of Defense. *A New Approach for Delivering Information Technology Capabilities in the Department of Defense*, November 2010.  
<https://acc.dau.mil/adl/en-US/412545/file/54776/New%20Acquisition%20Process OSD%2013744-10%20-%20804%20Report%20to%20Congress%202.pdf>

### **[Lapham 2010]**

Lapham, Mary Ann, Williams, Ray C., Hammons, Charles (Bud), Burton, Daniel, & Schenker, Alfred. *Considerations for Using Agile in DoD Acquisition* (CMU/SEI-2010-TN-002, ADA528647). Software Engineering Institute, Carnegie Mellon University, 2010.  
<http://www.sei.cmu.edu/library/abstracts/reports/10tn002.cfm>

### **[Lapham 2011]**

Lapham, Mary Ann, Miller, Suzanne, Adams, Lorraine, Brown, Nanette, Hackemack, Bart, Hammons, Charles (Bud), Levine, Linda, & Schenker, Alfred. *Agile Methods – Selected DoD Management and Acquisition Concerns* (CMU/SEI-2011-TN-002). Software Engineering Institute, Carnegie Mellon University, 2011.  
<http://www.sei.cmu.edu/library/abstracts/reports/11tn002.cfm>

<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved OMB No. 0704-0188</i>
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</p>			
1. AGENCY USE ONLY  (Leave Blank)	2. REPORT DATE  January 2014	3. REPORT TYPE AND DATES COVERED  Final	
4. TITLE AND SUBTITLE  Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs		5. FUNDING NUMBERS  FA8721-05-C-0003	
6. AUTHOR(S)  Mary Ann Lapham Michael Bandor Eileen Wrubel			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER  CMU/SEI-2013-TN-031	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES			
12A DISTRIBUTION/AVAILABILITY STATEMENT  Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE	
13. ABSTRACT (MAXIMUM 200 WORDS)  This technical note is the third in an SEI series on the adoption of lean and agile methods in the DoD. Agile topics in acquisition were introduced in CMU/SEI-2010-TN-002 and CMU/SEI-2011-TN-002. This technical note extends the topics covered into the evaluation and negotiation of technical proposals that reflect iterative development approaches that in turn leverage agile methods. This framework is intended for use by government program office personnel who seek to understand evaluation approaches in this context. The information and recommendations contained in this report result from observations of defense acquisition programs wherein contractors employed iterative methods such as Agile software development methodology (hereafter referred to as "agile"). Key questions for discussion with the contractor are provided, along with agile perspectives on why certain items will be defined differently depending on whether the contractor is using agile or iterative methods for software development. The intended audience for this paper includes any government personnel who need to support or participate in negotiations with contractors for changes to the contract that is in place to develop software using agile or iterative methods.			
14. SUBJECT TERMS  acquisition, agile, life cycle		15. NUMBER OF PAGES  54	
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT  Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE  Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT  Unclassified	20. LIMITATION OF ABSTRACT  UL